Santa: Faster Packet Delivery for Commonly Wished Replies

Florian Schmidt, Oliver Hohlfeld, René Glebke, Klaus Wehrle
Communication and Distributed Systems Group
RWTH Aachen University, Germany
{schmidt,hohlfeld,glebke,wehrle}@comsys.rwth-aachen.de

ABSTRACT

Increasing network speeds challenge the packet processing performance of networked systems. This can mainly be attributed to processing overhead caused by the split between the kernel-space network stack and user-space applications. To mitigate this overhead, we propose Santa, an application agnostic kernel-level cache of frequent requests. By allowing user-space applications to offload frequent requests to the kernel-space, Santa offers drastic performance improvements and unlocks the speed of kernel-space networking for legacy server software without requiring extensive changes.

CCS Concepts

•**Networks** → **Programming interfaces;**

1. THE QUEST FOR SPEED

Increasing line rates challenge the packet processing performance of current network stacks. These performance challenges can be attributed to two main overhead factors: *i)* memory allocations and copy operations, and *ii)* overheads by performing system calls and the required context switches. These costs particularly take effect at high line rates (e.g., multiple 10G interfaces) where many (small) requests need to be processed (e.g., DNS traffic). Thus, drastic increases in network line speeds significantly challenge packet processing in commodity hard- and software where CPU speeds do not scale with increasing line speeds.

One line of research addresses this problem by proposing kernel optimizations. Proposed optimizations involve to *i)* channelize processing [4], to *ii)* slim down socket buffers [2], or to *iii)* use batching to reduce overheads [5].

More radical approaches to tackle this challenge involve (partially or completely) bypassing the kernel by either *i)* offloading packet processing to specialized hardware (see e.g., [2, 7]) or by *ii)* shifting packet processing to user-land stacks (see e.g., [3, 6, 8]). The latter represents an active line of research that achieved drastic performance increases and lower CPU footprints by avoiding kernel based packet processing overheads [6]. These advances have proved to be useful for accelerating software switches [9], HTTP [3,6] and DNS servers [6].

SIGCOMM '15 August 17-21, 2015, London, United Kingdom
© 2015 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-3542-3/15/08.
DOI: http://dx.doi.org/10.1145/2785956.2790014

However, the drastic performance increases of user-land stacks also come at a price. Concretely, they require specialized microstacks running in dedicated applications to show their full potential. We complement this work by proposing a middle ground. That is, we trade off some of the potential performance increases of microstacks for the ability to rely on the well-established and full-featured network stack implementation of Linux. Instead of moving all network stack logic to the application, we enable applications to move parts of their logic into the stack. In contrast to earlier approaches that implement full server-*specific* logic in kernel space (e.g., khttpd [1]), we propose Santa, an application-*agnostic* mechanism allowing user-level server applications to offload requests to common replies to a kernel-level cache.

We exemplify the benefit of Santa by using a standard DNS server and offloading frequent DNS requests to our kernel cache. In this evaluation, we achieve drastic improvements in both response time (almost 2× faster) and throughput (up to 4× higher) even with a standard stack. Thus, Santa unlocks the speed of kernel-space networking for legacy server software without requiring extensive changes or specialized implementations. By this, we aim to pave the way for new packet processing pipelines and complement the ongoing discussion on user-level network stacks.

2. SANTA ARCHITECTURE

We show the Santa architecture in Figure 1 (c) and compare it to *(a)* a traditional network stack and *(b)* netmap [8] as a well-known approach to bypass the kernel for realizing user-land protocol stacks. The Santa architecture involves realizing a kernel-level cache that is instrumented by applications to offload replies to frequent requests. This way, we achieve performance increases by avoiding costly context switches to the user-space for frequent requests.

To enable reply offloading to the kernel cache, Santa extends the Linux kernel socket interface to enable applications to control the kernel-level cache. This cache control requires application modifications and comprises ways to install, update, and remove cache entries. Each cache entry includes a filter rule applied to incoming packets. If a rule matches, the kernel replies with a pre-cached response instead of forwarding the packet to the user-level process. Otherwise, the packet is handed over to the application via the socket interface, so that the server can create and send the reply.

Since the installed filter rules do not involve application

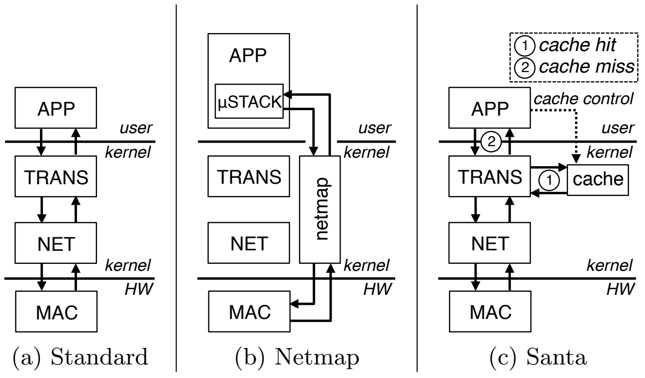

Figure 1: In a standard network stack, all packets are processed in both kernel and user space. Netmap bypasses the kernel to process packets in user space, requiring specialized microstacks. Santa provides a kernel cache to answer common requests without passing the packet to the application.

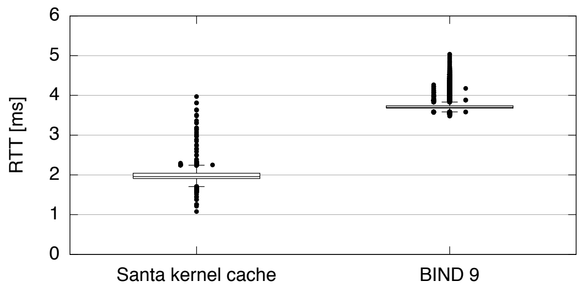

Figure 2: In a local network, the server's processing time forms a sizable part of the RTT. The Santa kernel cache provides significantly faster response times.

logic and match on packet contents, the Santa architecture is application agnostic. That is, every application can utilize the Santa API to offload packet processing to the kernel cache. In case that no cache items are installed, Santa behaves as a standard Linux kernel, i.e., all incoming packets get forwarded to the user-space process.

Santa is especially beneficial for applications that receive frequent requests that can be answered with pre-cached replies. A typical example application is DNS servers, where a small number of resource records is requested very frequently.

3. USE CASE EVALUATION: DNS SERVER

In our preliminary evaluation, we focus on accelerating BIND 9, a widely used DNS server.

Our experimental setup comprises one server (quad-core 3 GHz with 8 GB RAM) running the BIND 9 DNS server on top of our Santa kernel and four DNS clients with stock kernels generating DNS queries, connected via an Ethernet switch. Each machine is equipped with a 1 Gbit/s interface. We emulate a maximum-load scenario in which the clients send requests as fast as possible, saturating the link from the switch to the server. Figure 2 shows box plots of the round-trip time (RTT) distribution of requests, as seen by the client. Santa provides a significantly lower response time, with median values of 1.95 ms vs. 3.7 ms.

In a second set of experiments, we sent requests that hit the kernel cache with a certain probability (0%, 25%, 50%,

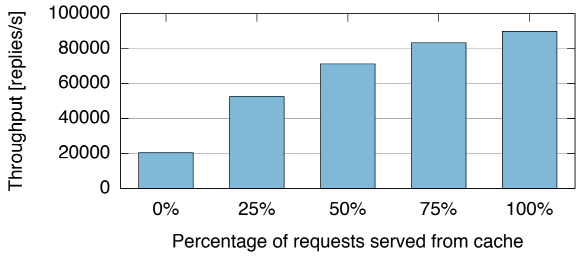

Figure 3: Serving requests with pre-cached replies drastically improves the request throughput of a server.

75%, 100%), and otherwise fell through to BIND. We then measured the number of requests that the server could process. Figure 3 shows that not only does Santa increase the throughput on our test machine by a factor of 4.4. It also shows throughput increases especially at lower cache hit rates: at 25% cache hits, throughput already increases by 258%, more than half the way to the maximum achieved increase. Thus, even in scenarios where our cache can only serve a small fraction of all requests, significant throughput improvements can be expected.

Concluding, our preliminary evaluation shows that drastic performance improvements are possible without clean-slate user-land network stack approaches.

4. FUTURE WORK

As next steps, we plan to evaluate the performance of Santa on real-world traces from major DNS servers, as well as to compare our solution with user-level network stacks. Furthermore, since Santa is currently limited to UDP, we plan to extend it to support TCP so that additional server applications can benefit from its caching.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation program 2014–2018 under grant agreement No. 644866 (SSICLOPS) and DFG CRC 1053 (MAKI). It reflects only the authors' views and the European Commission is not responsible for any use that may be made of the information it contains.

5. REFERENCES

[1] M. Bar. Kernel Korner: kHTTPd, a Kernel-Based Web Server. *Linux Journal*, 2000(76), Aug. 2000.

[2] S. Han, K. Jang, K. Park, and S. Moon. PacketShader: A GPU-accelerated software router. In *ACM SIGCOMM*, 2010.

[3] M. Honda, F. Huici, C. Raiciu, J. Araujo, and L. Rizzo. Rekindling Network Protocol Innovation with User-level Stacks. *SIGCOMM CCR*, 44(2):52–58, Apr. 2014.

[4] V. Jacobson and B. Felderman. Speeding up Networking. *linux.conf.au*, 2006.

[5] T. Marian, K. S. Lee, and H. Weatherspoon. NetSlices: Scalable Multi-Core Packet Processing in User-Space. In *ACM/IEEE ANCS*, 2012.

[6] I. Marinos, R. N. Watson, and M. Handley. Network Stack Specialization for Performance. In *ACM SIGCOMM*, 2014.

[7] I. Pratt and K. Fraser. Arsenic: A User-Accessible Gigabit Ethernet Interface. In *IEEE INFOCOM*, 2001.

[8] L. Rizzo. netmap: A Novel Framework for Fast Packet I/O. In *USENIX Security Symposium*, 2012.

[9] L. Rizzo and G. Lettieri. VALE, a Switched Ethernet for Virtual Machines. In *CoNEXT*, 2012.